Expressing Iterative Properties Logically in a Symbolic Setting

Carron Shankland[1], Jeremy Bryans[2], and Lionel Morel[3]

[1] Department of Computing Science and Mathematics, University of Stirling, UK
[2] Centre for Software Reliability, University of Newcastle, UK
[3] Verimag, France

Abstract. We present a logic for reasoning about state transition systems (LOTOS behaviours) which allows properties involving repeated patterns over actions and data to be expressed. The state transition systems are derived from LOTOS behaviours; however, the logic is applicable to any similar formalism. The semantics of the logic is given with respect to *symbolic* transition systems, allowing reasoning about data to be separated from reasoning about flow of control. Several motivational examples are included.
Keywords: LOTOS, formal verification, temporal logics, infinite state systems, symbolic representation.

1 Introduction

When describing a system formally it is often useful to be able to do so at different levels of abstraction. This enhances our confidence in the correctness of our mental model of the system, especially if the different descriptions can be related mathematically to each other. An obvious example is the use of modal and temporal logics to describe abstract properties of a system which can then be validated with respect to a more concrete, implementation focussed, description. For example, the use of PROMELA and CTL in the model checker SPIN [1].

In previous work we have established a symbolic framework for reasoning about Full LOTOS [2] based on *Symbolic Transition Systems* (STS) [3]. This new semantics for LOTOS allows behaviour to be expressed in a more compact, elegant fashion than is possible using the standard semantics [2]. In particular, this is a way of avoiding the state explosion caused by the use of data in LOTOS. A modal logic called FULL has been defined [4], but although this logic has the desirable theoretical property of adequacy with respect to bisimulation [5], it lacks the expressiveness required in many applications.

For example, consider the simple buffer which accepts data at gate `in` and outputs that data at gate `out` (in pseudo LOTOS, `B = in?x:Nat; out!x; B`). An abstract property of this buffer is *"after an* `in` *event, an* `out` *event occurs"*. Of course, it would be useful to express the repetition of this pattern: *"after the* `in` *action, the* `out` *action happens, and then repeat"*. We also want to express something about the data involved in the transaction *"if we input the value x*